

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-140652

(P2002-140652A)

(43) 公開日 平成14年5月17日 (2002.5.17)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード(参考)
G 0 6 F 19/00	1 1 0	G 0 6 F 19/00	1 1 0 5 B 0 4 9

審査請求 未請求 請求項の数 8 O L (全 7 頁)

(21) 出願番号 特願2000-333917(P2000-333917)

(22) 出願日 平成12年10月31日 (2000. 10. 31)

(71) 出願人 000003078

株式会社東芝

東京都港区芝浦一丁目1番1号

(72) 発明者 吉田 充伸

神奈川県川崎市幸区小向東芝町1番地 株

式会社東芝研究開発センター内

(74) 代理人 100058479

弁理士 鈴江 武彦 (外6名)

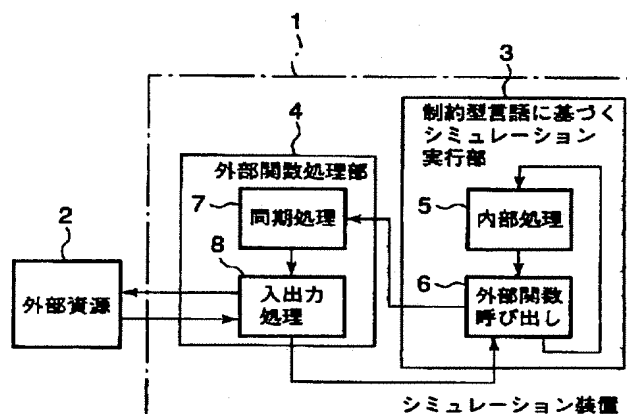
Fターム(参考) 5B049 AA02 DD00 EE41 FF03 FF09

(54) 【発明の名称】 シミュレーション方法、装置、及び記録媒体

(57) 【要約】

【課題】 制約型プログラミング言語に基づくシミュレーションを実行中に外部との同期を取りつつデータの入出力処理を行えるようにする。

【解決手段】 システムの時間軸上の挙動をシミュレートするシミュレーション装置において、シミュレーション実行部3は、制約型プログラミング言語で記述されたモデルのシミュレーションを実行し、当該シミュレーションの実行中に外部関数を呼び出すと共にその外部関数から返される戻り値に応じて当該シミュレーションの動作を変更する。外部関数処理部4は、シミュレーション実行部3からの呼び出しに応じ、シミュレーション実行部3と外部資源2との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータをシミュレーション実行部3に戻り値として返す外部関数を処理する。



1

## 【特許請求の範囲】

【請求項1】 システムの時間軸上の挙動をシミュレートするシミュレーション装置において、  
制約型プログラミング言語で記述されたモデルのシミュレーションを実行し、当該シミュレーションの実行中に外部関数を呼び出すと共にその外部関数から返される戻り値に応じて当該シミュレーションの動作を変更するシミュレーション実行手段と、  
前記シミュレーション実行手段からの呼び出しに応じ、前記シミュレーション実行手段と外部資源との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータを前記シミュレーション実行手段に戻り値として返す外部関数を処理する外部関数処理手段とを具備したことを特徴とするシミュレーション装置。

【請求項2】 前記外部関数処理手段は、前記シミュレーション実行手段が前記外部関数を呼び出す周期と実質的に等しい時間で当該外部関数を処理することを特徴とする請求項1記載のシミュレーション装置。

【請求項3】 前記外部関数処理手段は、前記シミュレーション実行手段からの呼び出しの種類に応じて、前記外部関数を処理する時間を変えることを特徴とする請求項1記載のシミュレーション装置。

【請求項4】 前記外部関数処理手段は、前記外部資源にアクセスする際にデータの書き込み及びデータの読み出しを要求することを特徴とする請求項1記載のシミュレーション装置。

【請求項5】 前記外部関数は、C言語で記述されたものであることを特徴とする請求項1記載のシミュレーション装置。

【請求項6】 前記制約型プログラミング言語は、HCC言語(Hybrid Concurrent Constraint Programming Language)であることを特徴とする請求項1記載のシミュレーション装置。

【請求項7】 システムの時間軸上の挙動をシミュレートするシミュレーション方法において、  
制約型プログラミング言語で記述されたモデルのシミュレーションの実行中に外部関数を呼び出し、  
前記外部関数により、前記モデルのシミュレーションと外部資源との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータを前記モデルのシミュレーションに反映させることを特徴とするシミュレーション方法。

【請求項8】 システムの時間軸上の挙動のシミュレーションを行うためのコンピュータプログラムを記録した、コンピュータにより読み取り可能な記録媒体において、前記コンピュータプログラムは、  
制約型プログラミング言語で記述されたモデルのシミュレーションの実行中に外部関数を呼び出し、  
前記外部関数により、前記モデルのシミュレーションと

2

外部資源との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータを前記モデルのシミュレーションに反映させる手順とを実行することを特徴とする記録媒体。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】 本発明は、シミュレーション装置に関し、特に、システム(系)の時間軸上の挙動をシミュレートするシミュレーション方法、装置、及び記録媒体に関する。

## 【0002】

【従来の技術】 時間依存のシミュレーション装置が外部資源とのデータの交換によってインタラクティブにシミュレーションを実行するためには、シミュレーション装置がファイル、共有メモリ、入出力ポートなどの外部資源にデータを出力する手段、外部資源からデータを入力する手段が必要になる。

【0003】 また、時間依存のシステムが外部資源とのデータの授受をある特定の時間に行うためには、シミュレーション装置内部で管理している内部時間と、タイマーなどの外部資源が管理している外部時間が一致している必要がある。そのため、このようなシミュレーション装置には外部資源と時間的な一致を取るための同期手段が必要になる。

【0004】 ところで、従来のシミュレーション装置の代表的な開発言語としては、アセンブラ、BASIC、Fortran、C、C++、JAVA(登録商標)などの言語がある。これらの言語を用いたシミュレーション装置で行われる処理について図8を用いて説明する。図8には、シミュレーション装置101と外部資源102とが示されている。シミュレーション装置101では、シミュレーションの内部処理103、外部資源102との時間的な整合性を取るための同期処理104、外部資源102との入出力を行う入出力処理105といった一連の処理が繰り返行われる。

【0005】 上述の言語は、標準関数やライブラリを用いることによって容易にファイルの読み書きや、メモリへのアクセス、外部通信ポートへのアクセスなどを実現することができ、外部資源との情報の交換が可能である。また、ハードウェアが持っているタイマー機能とライブラリや標準関数などを利用して実時間を取得することができるため、シミュレーション装置が内部で管理している時間との同期を取ることも容易である。

【0006】 このため、シミュレーション実行中に外部とのインタラクティブな入出力処理を行うシミュレーション装置を構築する場合には、上記言語自身が有している入出力インタフェース機能、時間管理機能を用いることができる。

【0007】 一方、別々に構成したモデルを読み込んで再構成可能なシステムの時間軸上の挙動をシミュレート

する装置としては、HCC言語(Hybrid Concurrent Constraint Programming Language)を用いたテキストベースで編集可能なインタプリタ、Mathematca言語のアドインツールやフロントエンド、Mapleフロントエンド、Mathcadフロントエンドなどがある。

【0008】これらの言語は、時間依存のシステムを微分方程式や恒等式などの拘束条件を用いて表記する機能を持っており、シミュレーションモデルを直感的に構築できるといった特徴や、モデルの再利用が容易であるといった特徴がある。ここでは、このような言語を制約型プログラミング言語（コンストレイントプログラミング言語）と呼ぶ。

【0009】また、こうした言語の中には、C言語などの他の言語を呼び出す機能を備えたものもある。この機能は、主に数学関数や複雑な論理演算などを高速に処理させるために用いられる。

【0010】

【発明が解決しようとする課題】しかし、制約型プログラミング言語は、それ自身が外部資源とデータを授受する機能を備えていないため、シミュレーション実行中に外部とのインタラクティブな入出力処理を行うシステムを構築することは不可能である。そのため、制約型プログラミング言語では、シミュレーションモデル作成時に、予め決められたシナリオどおりにシミュレーションを実行することしかできないという問題があった。

【0011】また、制約型プログラミング言語は、それ自身が実時間を取得する機能を有していないため、データの入出力処理を行うに際して外部との同期を取ることができないという問題があった。

【0012】本発明は上記実状に鑑みてなされたものであり、制約型プログラミング言語に基づくシミュレーションを実行中に外部との同期を取りつつデータの入出力処理を行うことのできるシミュレーション方法、装置、及び記録媒体を提供することを目的とする。

【0013】

【課題を解決するための手段】上記課題を解決するため、本発明によれば、システムの時間軸上の挙動をシミュレートするシミュレーション装置において、制約型プログラミング言語で記述されたモデルのシミュレーションを実行し、当該シミュレーションの実行中に外部関数を呼び出すと共にその外部関数から返される戻り値に応じて当該シミュレーションの動作を変更するシミュレーション実行手段と、前記シミュレーション実行手段からの呼び出しに応じ、前記シミュレーション実行手段と外部資源との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータを前記シミュレーション実行手段に戻り値として返す外部関数処理手段とを具備したことを特徴とするシミュレーション装置が提供される。

【0014】なお、前記構成において、前記外部関数処

理手段は、前記シミュレーション実行手段が前記外部関数を呼び出す周期と実質的に等しい時間で当該外部関数処理するものであってもよい。

【0015】また、前記構成において、前記外部関数処理手段は、前記シミュレーション実行手段からの呼び出しの種類に応じて、前記外部関数処理する時間を変えるものであってもよい。

【0016】また、前記構成において、前記外部関数処理手段は、前記外部資源にアクセスする際にデータの書き込み及びデータの読み出しを要求するものであってもよい。

【0017】また、前記構成において、前記外部関数は、C言語で記述されたものであってもよい。

【0018】また、前記構成において、前記制約型プログラミング言語は、HCC言語(Hybrid Concurrent Constraint Programming Language)であってよい。

【0019】また、本発明によれば、システムの時間軸上の挙動をシミュレートするシミュレーション方法において、制約型プログラミング言語で記述されたモデルのシミュレーションの実行中に外部関数を呼び出し、前記外部関数により、前記モデルのシミュレーションと外部資源との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータを前記モデルのシミュレーションに反映させることを特徴とするシミュレーション方法が提供される。

【0020】また、本発明によれば、システムの時間軸上の挙動のシミュレーションを行うためのコンピュータプログラムを記録した、コンピュータにより読み取り可能な記録媒体において、前記コンピュータプログラムは、制約型プログラミング言語で記述されたモデルのシミュレーションの実行中に外部関数を呼び出し、前記外部関数により、前記モデルのシミュレーションと外部資源との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータを前記モデルのシミュレーションに反映させる手順とを実行することを特徴とする記録媒体が提供される。

【0021】

【発明の実施の形態】以下、図面を参照して本発明の実施形態を説明する。図1は、本発明に係る第1の実施形態及び第2の実施形態に共通するシミュレーション装置の概念を説明するためのブロック図である。

【0022】同図には、シミュレーション装置1と外部資源2とが示されている。シミュレーション装置1は、システムの時間軸上の挙動をシミュレートするものであり、シミュレーション実行部3と外部関数処理4とを備えている。なお、外部資源2は、シミュレーション装置1の内部に含まれるように構成してもよい。

【0023】シミュレーション実行部3の中では、シミュレーションの内部処理5が行われ、外部資源2とのデータの入出力が必要な時点で外部関数の呼び出し6が行

われる。

【0024】すなわち、シミュレーション実行部3は、HCC言語などの制約型プログラミング言語（コンストレイントプログラミング言語）で記述された単一のシミュレーションモデル（以下、モデルと称する）ないしは別々に作成されたモデルのシミュレーションを実行し、当該シミュレーションの実行中に外部関数を呼び出すと共にその外部関数から返される戻り値に応じて当該シミュレーションの動作を変更するものである。

【0025】一方、外部関数処理部4の中では、図示のように外部関数の呼び出し6に応じてシミュレーション実行部3と外部資源2との時間的な整合性を取るための同期処理7が行われ、シミュレーション実行部3と外部資源2とのデータの入出力処理8が行われる。

【0026】すなわち、外部関数処理部4は、シミュレーション実行部3からの呼び出しに応じ、シミュレーション実行部3と外部資源2との時間的な整合性をとりつつ、当該外部資源2にアクセスしてデータを読み出してそのデータをシミュレーション実行部3に戻り値として返す外部関数を処理するものである。この関数は、上記モデルを記述したソースコード（プログラム）に対する外部関数としてC言語などの言語で記述されている。特に、外部関数処理部4は、シミュレーション実行部3が外部関数を呼び出す周期と実質的に等しい時間で当該外部関数を処理する。

【0027】このように、入出力処理および同期処理を行うルーチンを外部関数に持たせ、そのルーチンを拘束条件の一部として呼び出すことによって、外部資源へのアクセス及び同期を実現できるものとなっている。

【0028】図2は、本発明の第1の実施形態によるシミュレーション装置に関わるデータの流れを示すブロック図である。

【0029】同図に示されるように、シミュレーション装置1には、データ処理装置9およびシミュレーションデータ表示装置10が接続されている。データ処理装置9中に設けられるデータファイルは上述の外部資源2に相当するものである。なお、シミュレーションデータ表示装置10は、シミュレーション装置1の内部に含まれるように構成してもよい。

【0030】シミュレーション装置1は、上述の外部関数の機能を利用することにより、シミュレーションの実行中にデータ処理装置9のデータファイルにアクセスする。データ処理装置9は、シミュレーション装置1からのアクセスに応じ、データファイルから指定されたデータを読み出してシミュレーション装置1に送る。シミュレーション装置1は、読み出したデータを実行中のシミュレーションのプログラムに反映させる。そして、シミュレーション装置1は、上記データがプログラムに反映されたシミュレーションの実行結果をシミュレーションデータ表示装置10に表示させる。

【0031】なお、本実施形態によるシミュレーション装置1は、シミュレーションの実行中にデータ処理装置9のデータファイルからデータを読み出すことはできるが、データ処理装置9のデータファイルに対してデータを変更する（データを書き込む）ことはできない。

【0032】図3は、外部関数による外部との入出力処理を説明するための図である。同図には、シミュレーション実行部3によって実行されるモデルA及びモデルB、外部関数処理部4によって処理される外部関数、及びデータ処理装置9に保持されるデータファイルが示されている。

【0033】上記モデルA及びモデルBはそれぞれHCC言語で記述されており、予めソースコードとして指定された手順でのみ実行することができるものである。これらモデルA、モデルBは単体でも実行可能であるが、本実施形態ではシステム全体の挙動をシミュレートできるようにするため、図示のようにモデルAとモデルBとを統合（再結合）しておく。また、モデルBに記述されたHCC言語（ソースコード、プログラム）の中には、外部とのデータの入出力を可能とするために外部関数を呼び出しする部分が備えられる。なお、外部関数呼び出し部は、ここでは複数記述されているものとする。

【0034】一方、外部関数は、C言語で記述されており、ここでは一例としてgetData関数と称するものが使用される場合を考える。また、外部資源であるデータファイル中のデータは、テキスト形式で記述され、文字列の並びとして例えば0, 0, 0, 0, 0, 0, 0, 1, 0, 0…の順に記述されている。getData関数からの読み出し要求がある毎に、指定されたデータ（1又は0）がgetData関数に渡される。ここで、データは、外部から書き込みが可能な共有メモリ内に保存するようにしてもよい。

【0035】モデルA及びモデルBについてシミュレーションが実行され、処理がモデルB中の外部関数呼び出し部に差し掛かると、外部関数呼び出し部は、所定の引数を指定してgetData関数を呼び出す。getData関数は、同期をとるために、必要に応じて自身の実行時間の調整を行った後、外部のデータ処理装置に対して上記引数に対応するデータの読み出し要求を行い、データファイルから指定したデータを読み出す。getData関数は、読み出したデータに応じて戻り値を決定し、その戻り値を外部関数呼び出し部に返す。

【0036】なお、外部関数呼び出し部からシミュレーション中に外部関数を呼び出す周期と外部関数が処理されている時間が実質的に等しくなるように予め設定しておくものとする。この場合、外部関数呼び出し部からの呼び出しの種類に応じて外部関数を処理する時間を変えることも考えられる。

【0037】また、ファイルデータの値に応じた関数の戻り値を予め設定しておく（例えばデータの値が1であれば戻り値を1とし、データの値が0であれば戻り値を

0とする)。なお、ファイルデータは、他のアプリケーションが変更な状態で読み込むようにしてもよい。

【0038】本実施形態で使用するHCC言語では逐次的に処理が進められていくため、外部関数が処理されている途中にその他の処理が実行されることはなく、外部関数を用いて外部との同期を取ることが可能となる。

【0039】次に、図4のフローチャートを参照して、シミュレーション実行部3側で行われる動作を説明する。モデルBのプログラムは、シミュレーション実行中において例えば引数を“start”としてgetData関数を呼び出す(ステップA1)。この後、getData関数からの戻り値が1であるか0であるかを判定し(ステップA2)、戻り値が1であればモデルBの動作をオン状態にする(ステップA3)。一方、戻り値が0であれば何もしない。

【0040】次に、モデルBのプログラムは、例えば引数を“end”としてgetData関数を呼び出す(ステップA4)。この後、getData関数からの戻り値が1であるか0であるかを判定し(ステップA5)、戻り値が1であればモデルBの動作をオフ状態にする(ステップA6)。一方、戻り値が0であれば何もしない。

【0041】この後、シミュレーションを終了するか、あるいはシミュレーションをそのまま続行するかについて判別する(ステップA7)。続行する場合にはステップA1からの処理を繰り返し、そうでなければ本シミュレーションを終了する。

【0042】次に、図5のフローチャートを参照して、外部関数処理部4側で行われる動作を説明する。外部関数処理部4におけるgetData関数は、モデルBのプログラムから呼び出されると、指定された引数が“start”であるか否かを判定する(ステップB1)。引数が“start”であれば、getData関数は、これに対応する処理を行うに際して同期を保つには自身の実行時間の調整を行う必要があるものとみなして、例えば1秒待ち(ステップB2)、その後にデータ処理装置9のデータファイルから指定のデータを読み出す(ステップB3)。一方、引数が“start”でなければ(引数が“end”であれば)、getData関数は、待ち時間なしにデータ処理装置9のデータファイルから指定のデータを読み出す。

【0043】次に、getData関数は、データファイルから読み出したデータが0から1に変わった否かを判定する(ステップB4)。データが0から1に変わったのであれば、getData関数は戻り値に1をセットし(ステップB5)、一方、データが1から0に変わったのであれば、getData関数は戻り値に0をセットして、これをモデルBのプログラムに返す(ステップB6)。これにより、外部関数であるgetData関数の処理が終わる。

【0044】なお、本実施形態では、シミュレーション装置の実行中に例えばテキストエディタなどを用いて編集をした場合、編集結果に応じてシミュレーションの実

行状態が変化し、シミュレーションデータ表示装置10によってその変化を確認することができる。また、別のアプリケーションプログラムがそのデータを編集することによって、シミュレーションの実行状態を変えることも可能である。

【0045】このように、第1の実施形態によれば、HCC言語などの制約型プログラミング言語に基づくシミュレーションを実行中に外部のデータ処理装置9との同期を取りつつそのデータ処理装置9からデータを読み出してシミュレーションの動作を変更させることができる。

【0046】次に、第2の実施形態について説明する。図6は、本発明の第2の実施形態によるシミュレーション装置に関わるデータの流れを示すブロック図である。なお、以下では、上述の第1の実施形態と共通する要素には同一の符号を付し、その詳細な説明を省略する。

【0047】第1の実施形態(図2)ではデータ処理装置9のデータファイルからデータを読み出す場合を説明したが、本実施形態(図6)ではこれに加え、シミュレーション装置1からのアクセスによりデータ処理装置9のデータファイルにデータを書き込むことが可能である。なお、データ書き込みの場合も、データ読み出しの場合と同様に、外部関数を利用することによって同期および入出力処理が実現される。

【0048】なお、シミュレーション実行部3側で行われる動作については、図4に示したフローチャートとはほぼ同様であり、モデルBの外部関数呼び出し部から外部関数を呼び出す際にデータを渡す処理が加わるだけとなるため、ここではその説明を省略する。

【0049】次に、図7のフローチャートを参照して、外部関数処理部4側で行われる動作を説明する。外部関数処理部4におけるgetData関数は、モデルBのプログラムから呼び出されると、指定された引数が“start”であるか否かを判定する(ステップD1)。引数が“start”であれば、getData関数は、これに対応する処理を行うに際して同期を保つには自身の実行時間の調整を行う必要があるものとみなして、例えば1秒待ち(ステップD2)、その後にデータ処理装置9のデータファイルに指定のデータを書き込む(ステップD3)。一方、引数が“start”でなければ(引数が“end”であれば)、getData関数は、待ち時間なしにデータ処理装置9のデータファイルに指定のデータを書き込む。この後、getData関数は、データ処理装置9のデータファイルから指定のデータを読み出す(ステップD4)。

【0050】次に、getData関数は、データファイルから読み出したデータが0から1に変わったか否かを判定する(ステップD5)。データが0から1に変わったのであれば、getData関数は戻り値に1をセットし(ステップD6)、一方、データが1から0に変わったのであれば、getData関数は戻り値に0をセットして、これを

9

モデルBのプログラムに返す(ステップD7)。これにより、外部関数であるgetData関数の処理が終わる。

【0051】本発明の第2の実施形態によれば、シミュレーション実行中にシミュレーション実行部3からデータ処理装置9に中のデータファイルに書き込みを行ってデータを変更することができるとともに、データ処理装置9からデータを読み出してシミュレーションの動作を変更させることができるため、より複雑な処理が可能になり、多様なシミュレーションを行うことができる。

【0052】上記各実施形態で説明したシミュレーションの処理手順は、コンピュータプログラムとして、コンピュータにより読み取り可能な記録媒体に記録しておくことが可能である。そのコンピュータプログラムを必要時に読み出して実行することにより、シミュレーションを実行することができる。この場合、コンピュータプログラムの構成としては、制約型プログラミング言語で記述されたモデルのシミュレーションの実行中に外部関数を呼び出し、前記外部関数により、前記モデルのシミュレーションと外部資源との時間的な整合性をとりつつ、当該外部資源にアクセスしてデータを読み出してそのデータを前記モデルのシミュレーションに反映させる手順を含むようにすればよい。

【0053】なお、本発明は、上述した実施形態に限定されるものではなく、その要旨の範囲で種々変形して実施することが可能である。

【0054】例えば、上記実施形態ではHCC言語を使用した場合を例にとって説明したが、本発明はこれに限定されず、他の制約型プログラミング言語(Mathematica言語、Maple言語、Mathcad言語等)を使用してもよい。

【0055】

【発明の効果】以上詳記したように本発明によれば、制約型プログラミング言語に基づくシミュレーションを実

10

行中に外部との同期を取りつつデータの入出力処理を行うことのできるシミュレーション方法、装置、及び記録媒体を提供することができる。

【図面の簡単な説明】

【図1】本発明に係る第1の実施形態及び第2の実施形態に共通するシミュレーション装置の概念を説明するためのブロック図。

【図2】本発明の第1の実施形態によるシミュレーション装置に関わるデータの流れを示すブロック図。

【図3】同実施形態において外部関数による外部との入出力処理を説明するための図。

【図4】同実施形態においてシミュレーション実行部側で行われる動作を説明するためのフローチャート。

【図5】同実施形態において外部関数処理部側で行われる動作を説明するためのフローチャート。

【図6】本発明の第2の実施形態によるシミュレーション装置に関わるデータの流れを示すブロック図。

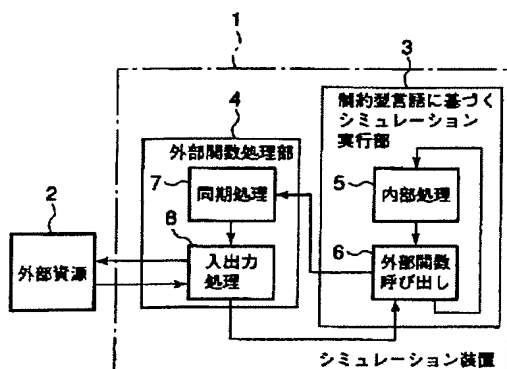
【図7】同実施形態において外部関数処理部側で行われる動作を説明するためのフローチャート。

【図8】従来のシミュレーション装置で行われる処理を説明するための図。

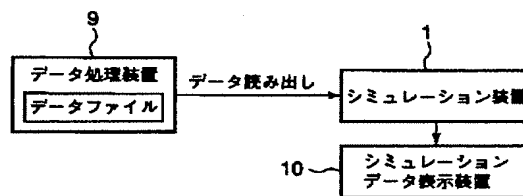
【符号の説明】

- 1…シミュレーション装置
- 2…外部資源
- 3…シミュレーション実行部
- 4…外部関数処理部
- 5…外部処理
- 6…外部関数呼び出し
- 7…同期処理
- 8…入出力処理
- 9…データ処理装置
- 10…シミュレーションデータ表示装置

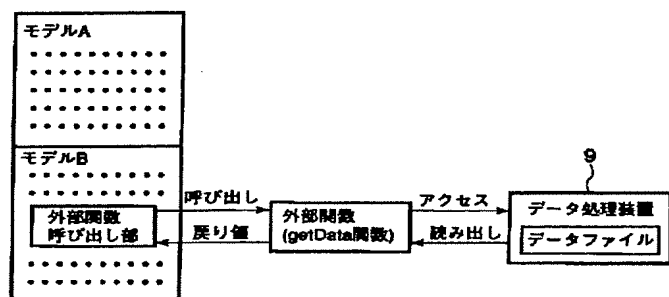
【図1】



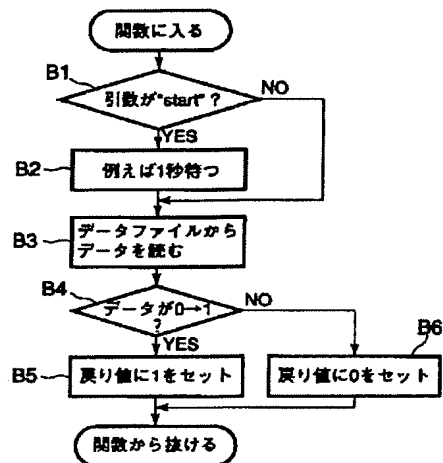
【図2】



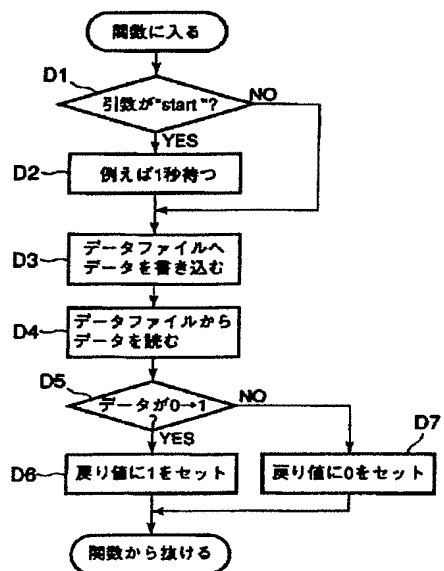
【図3】



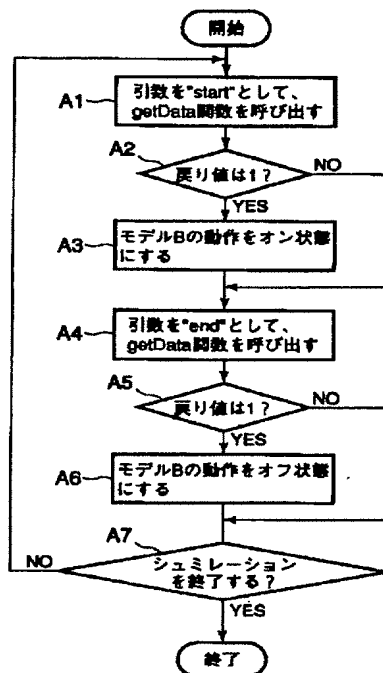
【図5】



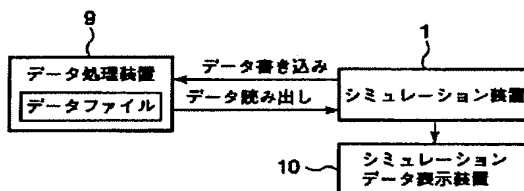
【図7】



【図4】



【図6】



【図8】

